REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE July 9, 1998	3. REPORT TYPE AND DATES COVERED Final report 5/15/95 - 5/14/98

4. TITLE AND SUBTITLE
Mechanisms for Scalable Object Sharing in MIMD Multiprocessing Systems

5. FUNDING NUMBERS
DAAH04-95-1-0323

6. AUTHOR(S)
James H. Anderson

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
University of North Carolina
Department of Computer Science
CB #3175, Sitterson Hall
Chapel Hill, NC 27599-3175

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 34162.17-MA-YIP

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In this project, object-sharing schemes for both real-time and non-real-time concurrent systems have been investigated. A particular focus has been uniprocessor and shared-memory multiprocessor systems with processors that are multiprogrammed (many processes executing on the same processor). Much of the work in this project has been concerned with *lock-free* and *wait-free* shared object implementations. Such implementations are not lock-based, and therefore are immune to performance problems associated with process preemptions in multiprogrammed systems. A variety of new algorithmic techniques for efficiently implementing concurrent objects have been developed and tested in this project. In addition, research has been conducted on schedulability tests for use in real-time systems in which the proposed object-sharing techniques are used.

14. SUBJECT TERMS
lock-free, wait-free, objects, real-time, databases

15. NUMBER IF PAGES
6

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

REPORT DOCUMENTATION PAGE (SF298)
(Continuation Sheet)

Mechanisms for Scalable Object Sharing in MIMD Multiprocessing Systems

Final Progress Report

James H. Anderson

July 1998

U.S. Army Research Office

Grant Number DAAH04-95-1-0323

Department of Computer Science
The University of North Carolina at Chapel Hill
Campus Box 3175
356 Sitterson Hall
Chapel Hill, North Carolina 27599-3175

Problem Studied

In this project, object-sharing schemes for both real-time and non-real-time concurrent systems have been investigated. A particular focus has been uniprocessor and shared-memory multiprocessor systems with processors that are multiprogrammed. In *multiprogrammed systems*, several processes may execute on the same processor. Processes on the same processor are scheduled for execution either by priority or by allocating a scheduling quantum. One of the main reasons for adopting a multiprogrammed execution model is that it enables problems to be solved without static constraints on the number of processes that may be employed. One price to be paid for this flexibility is having to deal with frequent process delays due to preemptions.

Much of the work in this project has been concerned with *lock-free* and *wait-free* shared object implementations. As explained in the original proposal and in previous interim reports, such implementations are not lock-based, and therefore are immune to performance problems associated with process preemptions in multiprogrammed systems. Most of the research conducted in this project has focused on two main goals: (i) to develop efficient algorithmic techniques for implementing concurrent objects, and (ii) to determine how to account for object-sharing overheads that arise when using such techniques in real-time schedulability tests.

Summary of Results

In most conventional wait-free and lock-free shared object implementations, performance does not scale well with an increase in either the number of processes sharing the object, or the size of the object. To deal with the former problem, we developed object implementations that incorporate lock-based synchronization techniques to limit the number of processes that can concurrently access an object [1]. Performance studies conducted by us have shown that, in multiprogrammed systems, the use of such techniques results in performance that is better than that of pure wait-free objects. To deal with the problem of implementing large objects, we developed wait-free and lock-free object constructions in which the state of the object is fragmented into smaller pieces that can be updated and managed more efficiently [15]. Performance studies conducted by us on a KSR-1 multiprocessor have shown that, for many common objects, these implementations perform significantly better than previous ones.

Our work on real-time object sharing has focused both on scheduling conditions for real-time tasks that share objects, and on algorithms for implementing such objects. For real-time applications on uniprocessors, we have shown through work on task scheduling that lock-free objects often outperform conventional lock-based schemes by a substantial margin [2,5,11,16]. The good performance of lock-free objects in this context is primarily due to the fact that they avoid priority inversions with no kernel support for object sharing (in contrast to lock-based schemes) and with low algorithmic overhead (for most common objects). We have validated these claims both formally, based on scheduling models, and experimentally, based on research involving a desktop videoconferencing system

We have also shown that it is possible to optimize wait-free algorithms in real-time multiprocessor systems by exploiting the inherent synchrony that exists in such systems [5,7,8,10,13]. We have evaluated the performance of these optimized wait-free implementations by conducting a variety of simulation experiments and by performing stress tests on a four-processor SGI Origin. These experiments indicate that wait-free objects implemented using the proposed techniques outperform other object-sharing schemes by a wide margin and lead to better schedulability in real-time systems.

We have also initiated new work on memory-resident real-time databases (RTDBs) [3,4,9,14]. RTDBs differ from conventional database systems in that transactions can have soft, firm, or hard deadlines. In most existing RTDBs, soft or firm deadlines are supported by modifying conventional database protocols to provide preferential treatment to high-priority transactions. In such systems, best-effort scheduling protocols are employed to minimize the number of transactions that miss their deadlines. In many military applications, however, certain critical transactions must be guaranteed to satisfy hard real-time constraints. For example, in an air defense system designed to neutralize enemy ballistic missiles, database updates may be triggered to track and identify potential threats and to issue appropriate counterstrikes; such updates *must* complete by their deadlines. Our work on RTDBs has focused on hard-deadline uniprocessor and multiprocessor systems. Our approach is to implement such transactions using highly-optimized lock-free and wait-free algorithms that execute at the user level. Since no underlying system support is needed for transactions, this allows RTDB functionality to be achieved in embedded applications without complicated protocols for avoiding priority inversion and deadlock, for supporting mode changes, and for handling transaction abort/recovery.

Publications

Journal Papers.

[1] J. Anderson and M. Moir, "Using Local-Spin *k*-Exclusion Algorithms to Improve Wait-Free Object Implementations", *Distributed Computing*, Volume 11, Number 1, pages 1-20, December 1997.

[2] J. Anderson, S. Ramamurthy, and K. Jeffay, "Real-Time Computing with Lock-Free Shared Objects", *ACM Transactions on Computer Systems*, Volume 15, Number 2, pages 134-165, May 1997.

Book Chapters.

[3] J. Anderson, R. Jain, and S. Ramamurthy, "Implementing Hard Real-Time Transactions on Multiprocessors", in *Real-Time Database and Information Systems: Research Advances*, Azer Bestavros and Victor Fay-Wolfe (eds.), Kluwer Academic Publishers, Norwell, Massachusetts, pages 247-260, September 1997.

[4] J. Anderson, S. Ramamurthy, M. Moir, and K. Jeffay, "Lock-Free Transactions for Real-Time Systems", in *Real-Time Database Systems: Issues and Applications*, A. Bestavros, K.J. Lin, and S.H. Son, (eds.), Kluwer Academic Publishers, Norwell, Massachusetts, pages 215-234, May 1997.

Conference Papers.

[5] J. Anderson, R. Jain, and K. Jeffay, "Efficient Object Sharing in Quantum-Based Real-Time Systems", submitted to the 19th IEEE Real-Time Systems Symposium, 23 pages, May 1998.

[6] K. Jeffay, F. Donelson Smith, A. Moorthy, and J. Anderson, "Proportional Share Scheduling of Operating System Services for Real-Time Applications", submitted to the 19th IEEE Real-Time Systems Symposium, 24 pages, May 1998.

[7] J. Anderson, R. Jain, and D. Ott, "Wait-Free Synchronization in Quantum-Based Multiprogrammed Systems", to be presented at the 12th International Symposium on Distributed Computing, 21 pages, September 1998.

[8] J. Anderson, R. Jain, and S. Ramamurthy, "Wait-Free Object-Sharing Schemes for Real-Time Uniprocessors and Multiprocessors", *Proceedings of the 18th IEEE Real-Time Systems Symposium*, IEEE Computer Society Press, pages 111-122, December 1997.

[9] J. Anderson, R. Jain, and S. Ramamurthy, "Implementing Hard Real-Time Transactions on Multiprocessors", *Proceedings of the Second International Workshop on Real-Time Databases*, pages 247-260,

September 1997.

[10] J. Anderson, S. Ramamurthy, and R. Jain, "Implementing Wait-Free Objects on Priority-Based Systems", *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, ACM, New York, pages 229-238, August 1997.

[11] J. Anderson and S. Ramamurthy, "A Framework for Implementing Objects and Scheduling Tasks in Lock-Free Real-Time Systems", *Proceedings of the 17th IEEE Real-Time Systems Symposium*, IEEE Computer Society Press, pp. 94-105, December 1996.

[12] Mark Moir and Juan Garay, "Fast, Long-Lived Renaming Improved and Simplified", *Proceedings of the 10th International Workshop on Distributed Algorithms*, Lecture Notes in Computer Science 1151, Springer-Verlag, pp. 287-303, October 1996.

[13] S. Ramamurthy, M. Moir, and J. Anderson, "Real-Time Object Sharing with Minimal System Support", *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing*, ACM, New York, pp. 233-242, May 1996.

[14] J. Anderson, S. Ramamurthy, M. Moir, and K. Jeffay, "Lock-Free Transactions for Real-Time Systems", *Proceedings of the First International Workshop on Real-Time Databases: Issues and Applications*, pp. 107-114, March 1996.

[15] J. Anderson and M. Moir, "Universal Constructions for Large Objects", *Proceedings of the Ninth International Workshop on Distributed Algorithms*, Lecture Notes in Computer Science 972, Springer-Verlag, pages 168-182, September 1995.

[16] J. Anderson, S. Ramamurthy, and K. Jeffay "Real-Time Computing with Lock-Free Shared Objects (Extended Abstract)", *Proceedings of the 16th IEEE Real-Time Systems Symposium*, IEEE Computer Society Press, pages 28-37, December 1995.

Participating Scientific Personnel

Faculty.

James H. Anderson, PI, Associate Professor.

Graduate Students.

Rohit Jain (partially funded by other sources).
Mark Moir (partially funded by other sources).
Srikanth Ramamurthy.

Honors/Awards/Degrees.

Alfred P. Sloan Research Fellowship, February 1996, Prof. Anderson.

Ph.D. awarded, August 1996, Mark Moir. Dissertation title: *Efficient Object Sharing in Shared-Memory Multiprocessors*.

REPORT DOCUMENTATION PAGE (SF298)
(Continuation Sheet)

Ph.D. awarded, December 1997, Srikanth Ramamurthy. Dissertation title: *A Lock-Free Approach to Object Sharing in Real-Time Systems*.

M.S. awarded, May 1998, Rohit Jain.

Report of Inventions (by Title Only)

Wait-free algorithms for implementing large shared objects.

Scalable shared object algorithms for large multiprogrammed systems.

Algorithms for implementing lock-free transactions on priority-scheduled real-time uniprocessor systems.

Algorithms for implementing wait-free objects on priority- and quantum-scheduled uniprocessor and multiprocessor systems.

Algorithms for implementing wait-free transactions on priority-scheduled real-time multiprocessor systems.

Scheduling conditions for lock-free, real-time uniprocessor systems.

Scheduling conditions for wait-free, real-time uniprocessor and multiprocessor systems.